USBizi User Manual

Rev. 4.11 August 3, 2010 User Manual

USBizi Chipset LQFP144

USBizi Chipset LQFP100

Document Information

Information	Description
Abstract	This document covers information about USBizi Chipset, specifications, tutorials and references.

Revision History	
Date	**Modification**
07/21/10	Updated information for NETMF 4.1
07/12/10	Added more pins
05/18/10	Added more pins
03/05/10	Various modifications
03/03/10	First version

Table of Contents

1.Introduction....4
1.1.What is Microsoft .NET Micro Framework (NETMF)?....4
1.2.NETMF - Porting vs. Using....4
1.3.GHI's .NET Micro Framework Based Solutions....5
1.4.What is USBizi Chipset?....5
1.5.Block Diagram....6
1.6.USBizi Chipset Key Features....6
1.7.Example Applications....6
2.Pin-Out Description....7
2.1.USBiziTM LQFP 144 Pin-out Table....8
2.2.USBiziTM LQFP 100 Pin-out Table....11
3.USBizi On boot up....14
3.1.GHI Boot Loader vs. USBizi Firmware....14
3.2.Boot up Modes Settings and Access Interface Drivers....15
4.GHI Boot Loader....16
4.1.GHI Boot Loader Commands....16
4.2.USBizi Firmware Update....17
4.3.Managed (User) Application in Production Stage with Copy Protection....18
5.USBizi Firmware....19
5.1.Getting Started with USBizi....19
All you need to start up....19
First Power-up....19
Adding GHI NETMF Library....23
5.2.Recommended Hardware and Tutorials for USBizi....25
6.USBizi Features....27
6.1.Application Flash/RAM....27
6.2.Debugging Interface (Access Interface)....27
6.3.Digital Inputs/Outputs....28
6.4.Serial Peripherals....28
Serial Port (UART)....28
SPI....28
I2C....29
CAN....29
One-wire Interface....29
6.5.Networking....29
Ethernet....29
6.6.Graphics....29
6.7.USB Client (Device)....30
USB Cable Connection Detection....31
6.8.USB Host and Supported Class Drivers....32
6.9.Storage Devices (SD, USB MS) / File System....32
SD/MMC Memory....32
USB Mass Storage....33
6.10.Analog Inputs/Outputs....33
6.11.PWM....33
6.12.Output Compare....33
6.13.Battery RAM....33
6.14.Power Control / Hibernate....34
Power Control....34
Hibernate....34
6.15.Real Time Clock....34
6.16.Processor Register Access....34
6.17.Managed Application Protection....34
6.18.Watchdog....34
7.Advanced Users....35
8.USBizi Design Consideration....35
Legal Notice....36
Licensing....36
Disclaimer....36

1. Introduction

1.1. What is Microsoft .NET Micro Framework (NETMF)?

Microsoft .NET Micro Framework is a lightweight implementation of .NET Framework. It focuses on the specific requirements of resource-constrained embedded systems. Supporting development in C# and debugging on an emulation or the device, both using Microsoft's Visual Studio. The .NET Micro Framework is also open source, released under the Apache 2.0 license and completely free.

Developers already experienced with .NET and Visual Studio can take advantage of their skills immediately reducing the learning curve. The actual C# application development process is completely shielded from the low-level design details of the hardware platform. Combining the benefits with off-the-shelf, low-cost, network-enabled embedded systems creates a rapid product development solution.

1.2. NETMF - Porting vs. Using

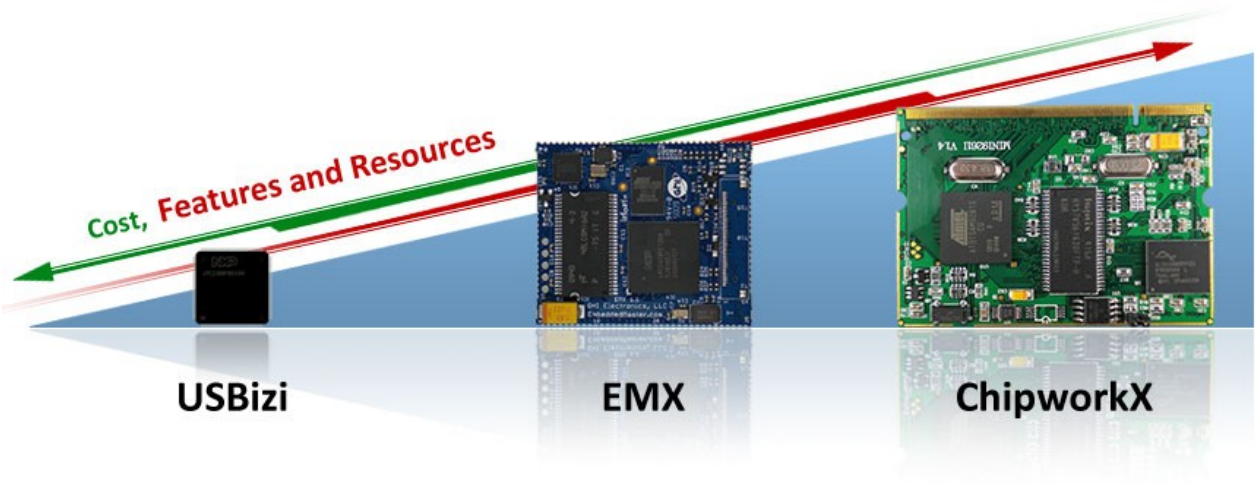
There are two sides to working with NETMF, porting it and using it. For example, writing a JAVA game on a cell phone is much easier than porting the JAVA virtual machine (JVM) to the phone. The phone manufacturer did all the hard work of porting JAVA to their phone allowing the game programmers to use it with ease. NETMF works the same way, porting is not easy but using it is effortless.

NETMF can be split into two major components, the core (CLR) and HAL (Hardware Access Layer). The core libraries are made so they are hardware independent. Usually, no modifications are needed on the core libraries. A developer porting NETMF for a hardware platform will need to make the HAL to handle interfacing the hardware control to upper layers.

According to GHI's experience with NETMF porting, it is not feasible to work on porting NETMF to your new hardware in case you are targeting medium or low quantities annually (less than 100,000 units). A faster-to-market option is by using one of the available OEM modules/chipsets. These OEM devices have everything you need built in the hardware and software.

1.3. GHI's .NET Micro Framework Based Solutions

With GHI Electronics, you're getting an experienced partner that offers a wide range of .NET Micro Framework hardware and software capabilities using the various drop-in modules/chipsets such as ChipworkX, Embedded Master, EMX and USBizi. In addition, our free unlimited support is available to assist you at any point. New features and fixes come seamlessly to your product at no cost to you.

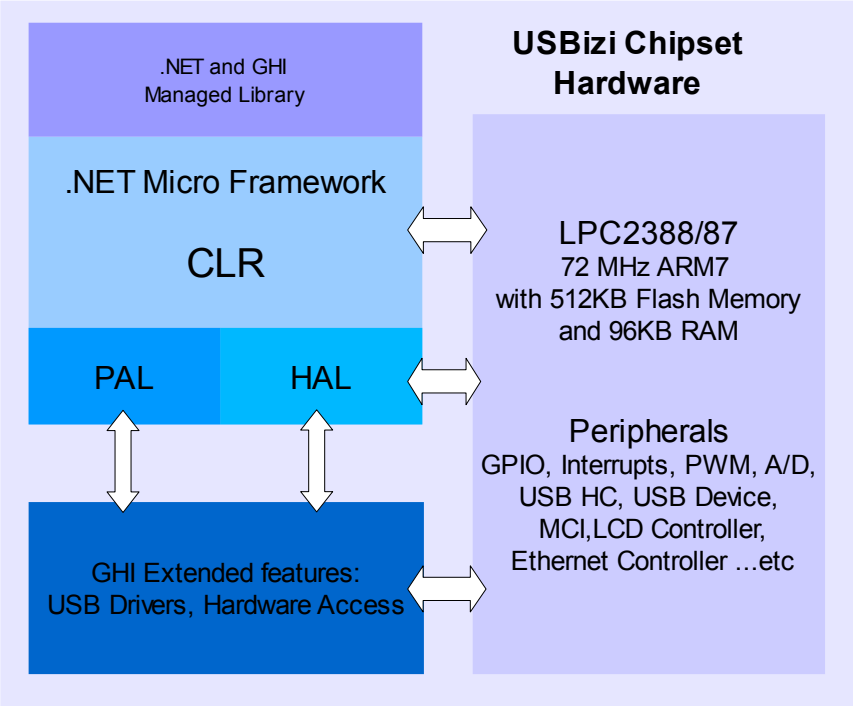


On top of the great features that .NET Micro Framework provides, such as Ethernet, graphics and touch screen, GHI solutions has additional exclusive features such as USB host, PPP (GPRS/3G), database and native code runtime libraries (RLP). All these exclusive features are included at no extra cost to you.

1.4. What is USBizi Chipset?

USBizi Chipset, pronounced as *USB easy*, is an ARM7 microcontroller from NXP (LPC2388 or LPC2387) with a special firmware developed to host .NET Micro Framework with various HAL drivers. With this sophisticated combination, a developer can easily control this microcontroller IOs and interfaces such as SPI, UART(Serial Port) and I2C with simple unified managed code (C# code) and enjoy the wide various higher level services offered by NETMF such as file system access. Not to mention GHI's extended features such as PWM, ADC, DAC and Full USB host stack. Everything implemented on a single-chip making USBizi the smallest and most inexpensive .NET Micro Framework device, and still implements unique features like USB host.

1.5. Block Diagram



1.6. USBizi Chipset Key Features

- Microsoft .NET Micro Framework V4
- 72Mhz 32-bit Processor
- 96KB RAM
- 0.5MB FLASH
- Embedded USB Host(144 pin package only)
- Embedded USB Client (Device)
- 71 GPIO
- 35 Interrupt Inputs
- 2 SPI (8/16bit)
- I2C
- 4 UART
- 2 CAN Channels
- 8 10-bit Analog Inputs (6 on 100pin package)
- 10-bit Analog Output
- 4-bit SD/MMC interface
- 6 PWM
- 100 mA everything enabled
- 200 uA Hibernate Modes
- -40ºC to +85ºC Operational
- RoHS Lead Free

1.7. Example Applications

- Data logger.
- MP3 player.
- Measurement tool or tester.
- Robotics.
- GPS navigation.
- Medical instrumentation.
- Industrial automation devices.

2. Pin-Out Description

USBizi Chipset is based on LPC2388 or LPC2387 72Mhz ARM7 32-bit processor. The processor has a wide range of peripherals that adds a lot of functions and features to USBizi such as PWM, GPIO, USB HC ...etc.

Most signals on USBizi are multiplexed to offer more than one function for every pin. It is up to the developer to select which one of the functions to use. GHI drivers and .NET Micro Framework does checking to make sure the user is not trying to use two functions on the same pin. The developer should still understand what functions are multiplexed so there is no conflict. For example, analog channel 3 (ADC3) and the analog output (AOUT) are on the same pin IO22. Either function can be used but not both of them simultaneously. Visit Advanced Users section.

- Pins not mentioned cannot be accessed directly, however, they can be used using the ***Register*** class provided by the GHI native library. Visit Advanced Users section.
- The schematics of USBizi boards should be used as a reference design. The boards are USBizi Development System, FEZ Domino and FEZ Mini.
- Advanced details on oscillator and power tolerance can be found in the LPC2388 datasheet from NXP website.
- Digital I/O pins are named IOxx, where xx is an assigned number.

2.1. USBizi™ LQFP 144 Pin-out Table

USBizi LQFP144 chipset is based on LPC2388 from NXP.

	Name			* Interrupt capable input.
No.	**LPC2388 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
8	P0.26	IO22*	ADC3/ DAC	ADC3 (10-Bit Analog to Digital Input) or DAC (Digital to Analog Output)
10	P0.25	IO24*	ADC2	ADC2 (10-Bit Analog to Digital Input)
11	P0.24	IO26*	ADC1	ADC1 (10-Bit Analog to Digital Input)
13	P0.23	IO28*	ADC0	ADC0 (10-Bit Analog to Digital Input)
14	VDDA			Connect to 3.3 volt source
15	VSSA			Connect to Ground
17	VREFA			Connect to 3.3 volt source
18	VDD_1			Connect to 3.3 volt source
22	VSS1			Connect to Ground
23	RTCX1			Connect to real-time crystal circuit pin 1
24	RESET#			Hardware reset signal, Reset state is on Low
25	RTCX2			Connect to real-time crystal circuit pin 2
26	ALARM			The alarm pin is an RTC controlled output. This is a 1.8 V pin.
27	VBAT			Connect to 3.3 volt back up battery to keep the real-time clock running.
28	P1.31	IO32	ADC5	ADC5 (10-Bit Analog to Digital Input)
29	P0.12	IO30*	ADC6	ADC6 (10-Bit Analog to Digital Input)
30	P1.30	IO29	ADC4	ADC4 (10-Bit Analog to Digital Input)
31	XTAL1			Connect to the system's 12MHz. crystal pin1
32	P0.13	IO27*	ADC7	ADC7 (10-Bit Analog to Digital Input)
33	XTAL2			Connect to the systems 12MHz. crystal pin1
34	P0.28	IO31*	I2C	**(open drain pin)** I2C Interface SCL
35	P0.27	IO33*	I2C	**(open drain pin)** I2C Interface SDA
36	USBD+ USB Client Feature			USB positive data line, USB (access) debugging interface and for the USB client feature.
37	USBD- USB Client Feature			USB negative data line, USB (access) debugging interface and for the USB client feature.
38	P3.26	IO69	N/A	General purpose digital I/O
39	P3.25	IO68	N/A	General purpose digital I/O
39	P3.24	IO70	N/A	General purpose digital I/O
41	VDD1			Connect to 3.3 volt source
42	USBD+ USB Host Feature			USB positive data line of the USB hosting feature.
43	USBD- USB Host Feature			USB negative data line of the USB hosting feature.
44	VSS2			Connect to Ground
46	P1.18	IO35	PWM1	PWM1 (Pulse Width Modulation Output)
47	P1.19	IO34	N/A	General purpose digital I/O
48	USB_CON USB Client Feature			Reports the USB cable attachment to the PC USB Host (refer to the reference design schematic for the required circuit).
49	P1.20	IO25	PWM2	PWM2 (Pulse Width Modulation Output)
50	P1.21	IO23	PWM3	PWM3 (Pulse Width Modulation Output)
51	P1.22	IO61	USB_PWR_RD	Related to USB host feature (refer to the reference design schematic for connection)
53	P1.23	IO62	N/A	General purpose digital I/O
54	P1.24	IO63	N/A	General purpose digital I/O
56	P1.25	IO64	N/A	General purpose digital I/O

	Name			* Interrupt capable input.
No.	**LPC2388 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
57	P1.26	IO65	N/A	General purpose digital I/O
59	VSS3			Connect to Ground
60	VDD_2			Connect to 3.3 volt source
61	P1.27	IO21	N/A	General purpose digital I/O
62	VDD2			Connect to 3.3 volt source
63	P1.28	IO66	N/A	General purpose digital I/O
64	P1.29	IO67	N/A	General purpose digital I/O
65	VSS4			Connect to Ground
66	P0.0	IO19*	CAN1	RD CAN Channel 1 Data Receive pin (In)
67	P0.1	IO15*	CAN1	TD CAN Channel 1 Data Transmit pin (Out)
69	P0.10	IO39*	N/A	Only general purpose digital I/O
70	P0.11	IO38*	N/A	Only general purpose digital I/O
71	P2.13	IO50*	SD_DAT3	SD card 4-bit data bus, data line no. 3
73	P2.12	IO49*	SD_DAT2	SD card 4-bit data bus, data line no. 2
75	P2.11	IO48*	SD_DAT1	SD card 4-bit data bus, data line no. 1
76	P2.10	IO0*	**BL#**	General purpose digital I/O. On power up, this pin is used to access GHI boot loader when Low (refer to GHI boot loader section).This pin is high (no loader) if not connected.
77	VDD3			Connect to 3.3 volt source
79	VSS5			Connect to Ground
80	P0.22	IO47*	SD_DAT0	SD card 4-bit data bus, data line no. 0
82	P0.21	IO46*	MCIPWR	Memory card (SD/MMC) power enable signal (refer to the reference design schematic for connection).
83	P0.20	IO45*	SD_CMD	SD card 4-bit data bus, command line
85	P0.19	IO44*	SD_CLK	SD card 4-bit data bus, clock line
86	P0.18	IO41*	SPI1	SPI master bus interface MOSI signal (Master Out Slave In) for SPI1
87	P0.17	IO40*	SPI1	SPI master bus interface MISO signal (Master In Slave Out)for SPI1
89	P0.15	IO42*	SPI1	SPI master bus interface SCK signal (Clock)for SPI1
90	P0.16	IO43*	N/A	General purpose digital I/O
92	P2.9	IO37*	COM3	Serial port (UART) RXD receive signal (In) for COM3
93	P2.8	IO36*	COM3	Serial port (UART) TXD transmit signal (Out) for COM3
95	P2.7	IO11*	COM2	Serial port (UART) RTS signal for COM2
96	P2.6	IO9*	N/A	Only general purpose digital I/O
97	P2.5	IO7*	PWM6	PWM6 (Pulse Width Modulation Output)
99	P2.4	IO2*	PWM5	PWM5 (Pulse Width Modulation Output)
100	P2.3	IO4*	PWM4/**MODE**	PWM4 (Pulse Width Modulation Output). On power up, this pin is used to select the access interface for GHI boot loader and debugging, between USB (High) or COM1(Low). (refer to GHI boot loader section). This pin is high (select USB) if not connected.
102	VDD4			Connect to 3.3 volt source
103	VSS6			Connect to Ground
105	P2.2	IO1*	COM2	Serial port (UART) CTS signal for COM2
106	P2.1	IO3*	COM2	Serial port (UART) RXD receive signal (In) for COM2
107	P2.0	IO5*	COM2	Serial port (UART) TXD transmit signal (Out) for COM2
109	P0.9	IO6*	SPI2	SPI master bus interface MOSI signal (Master Out Slave In) for SPI2
111	P0.8	IO8*	SPI2	SPI master bus interface MISO signal (Master In Slave Out)for SPI2
112	P0.7	IO10*	SPI2	SPI master bus interface SCK signal (Clock)for SPI2
113	P0.6	IO12*	N/A	General purpose digital I/O
114	VDD5			Connect to 3.3 volt source
115	P0.5	IO14*	CAN2	TD CAN Channel 2 Data Transmit pin (Out)

	Name			* Interrupt capable input.
No.	**LPC2388 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
116	P0.4	IO16*	CAN2	RD CAN Channel 2 Data Receive pin (In)
117	VSS7			Connect to Ground
118	P4.28	IO13	COM4	Serial port (UART) TXD transmit signal (Out) for COM4
119	VSS8			Connect to Ground
121	VDD_3			Connect to 3.3 volt source
122	P4.29	IO17	COM4	Serial port (UART) RXD receive signal (In) for COM4
123	P1.17	IO60	N/A	General purpose digital I/O
125	P1.16	IO59	N/A	General purpose digital I/O
126	P1.15	IO58	N/A	General purpose digital I/O
128	P1.14	IO57	N/A	General purpose digital I/O
129	P1.10	IO56	N/A	General purpose digital I/O
131	P1.9	IO55	N/A	General purpose digital I/O
132	P1.8	IO54	N/A	General purpose digital I/O
133	P1.4	IO53	N/A	General purpose digital I/O
135	P1.1	IO52	N/A	General purpose digital I/O
136	P1.0	IO51	N/A	General purpose digital I/O
138	VDD6			Connect to 3.3 volt source
139	VSS9			Connect to Ground
141	P0.2	IO18*	COM1	Serial port (UART) TXD transmit signal (Out) for COM1
142	P0.3	IO20*	COM1	Serial port (UART) RXD receive signal (In) for COM1

2.2. USBizi™ LQFP 100 Pin-out Table

USBizi LQFP100 chipset is based on LPC2387 from NXP.

	Name			* Interrupt capable input.
No.	**LPC2387 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
6	P0.26	IO22*	ADC3/ DAC	ADC3 (10-Bit Analog to Digital Input) or DAC (Digital to Analog Output)
7	P0.25	IO24*	ADC2	ADC2 (10-Bit Analog to Digital Input)
8	P0.24	IO26*	ADC1	ADC1 (10-Bit Analog to Digital Input)
9	P0.23	IO28*	ADC0	ADC0 (10-Bit Analog to Digital Input)
10	VDDA			Connect to 3.3 volt source
11	VSSA			Connect to Ground
12	VREFA			Connect to 3.3 volt source
13	VDD_1			Connect to 3.3 volt source
15	VSS1			Connect to Ground
16	RTCX1			Connect to real-time crystal circuit pin 1
17	RESET#			Hardware reset signal, Reset state is on Low
18	RTCX2			Connect to real-time crystal circuit pin 2
19	VBAT			Connect to 3.3 volt back up battery to keep the real-time clock running.
20	P1.31	IO32	ADC5	ADC5 (10-Bit Analog to Digital Input)
21	P1.30	IO29	ADC4	ADC4 (10-Bit Analog to Digital Input)
22	XTAL1			Connect to the system's 12MHz. crystal pin1
23	XTAL2			Connect to the systems 12MHz. crystal pin1
24	P0.28	IO31*	I2C	**(open drain pin)** I2C Interface SCL
25	P0.27	IO33*	I2C	**(open drain pin)** I2C Interface SDA
26	P3.26	IO69	N/A	General purpose digital I/O
27	P3.25	IO68	N/A	General purpose digital I/O
28	VDD1			Connect to 3.3 volt source
29	USBD+ USB Client Feature			USB positive data line, USB (access) debugging interface and for the USB client feature.
30	USBD- USB Client Feature			USB negative data line, USB (access) debugging interface and for the USB client feature.
31	VSS2			Connect to Ground
32	P1.18	IO35	PWM1	PWM1 (Pulse Width Modulation Output)
33	P1.19	IO34	N/A	General purpose digital I/O
34	P1.20	IO25	PWM2	PWM2 (Pulse Width Modulation Output)
35	P1.21	IO23	PWM3	PWM3 (Pulse Width Modulation Output)
36	P1.22	IO61	N/A	General purpose digital I/O
37	P1.23	IO62	N/A	General purpose digital I/O
38	P1.24	IO63	N/A	General purpose digital I/O
39	P1.25	IO64	N/A	General purpose digital I/O
40	P1.26	IO65	N/A	General purpose digital I/O
41	VSS3			Connect to Ground
42	VDD_2			Connect to 3.3 volt source
43	P1.27	IO21	N/A	General purpose digital I/O
44	P1.28	IO66	N/A	General purpose digital I/O
45	P1.29	IO67	N/A	General purpose digital I/O
46	P0.0	IO19*	CAN1	RD CAN Channel 1 Data Receive pin (In)
47	P0.1	IO15*	CAN1	TD CAN Channel 1 Data Transmit pin (Out)

	Name			* Interrupt capable input.
No.	**LPC2387 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
48	P0.10	IO39*	COM3	Serial port (UART) TXD transmit signal (Out) for COM3
49	P0.11	IO38*	COM3	Serial port (UART) RXD receive signal (In) for COM3
50	P2.13	IO50*	SD_DAT3	SD card 4-bit data bus, data line no. 3
51	P2.12	IO49*	SD_DAT2	SD card 4-bit data bus, data line no. 2
52	P2.11	IO48*	SD_DAT1	SD card 4-bit data bus, data line no. 1
53	P2.10	IO0*	**BL#**	General purpose digital I/O. On power up, this pin is used to access GHI boot loader when Low (refer to GHI boot loader section) This pin is high (no loader) if not connected.
54	VDD2			Connect to 3.3 volt source
55	VSS4			Connect to Ground
56	P0.22	IO47*	SD_DAT0	SD card 4-bit data bus, data line no. 0
57	P0.21	IO46*	MCIPWR	Memory card (SD/MMC) power enable signal (refer to the reference design schematic for connection).
58	P0.20	IO45*	SD_CMD	SD card 4-bit data bus, command line
59	P0.19	IO44*	SD_CLK	SD card 4-bit data bus, clock line
60	P0.18	IO41*	SPI1	SPI master bus interface MOSI signal (Master Out Slave In) for SPI1
61	P0.17	IO40*	SPI1	SPI master bus interface MISO signal (Master In Slave Out) for SPI1
62	P0.15	IO42*	SPI1	SPI master bus interface SCK signal (Clock)for SPI1
63	P0.16	IO43*	N/A	General purpose digital I/O
64	USB_CON USB Client Feature			Reports the USB cable attachment to the PC USB Host (refer to the reference design schematic for the required circuit).
65	P2.8	IO36*	N/A	Only general purpose digital I/O.
66	P2.7	IO11*	COM2	Serial port (UART) RTS signal for COM2
67	P2.6	IO9*	N/A	General purpose digital I/O
68	P2.5	IO7*	PWM6	PWM6 (Pulse Width Modulation Output)
69	P2.4	IO2*	PWM5	PWM5 (Pulse Width Modulation Output)
70	P2.3	IO4*	PWM4/ **MODE**	PWM4 (Pulse Width Modulation Output), On power up, this pin is used to select the communication interface for GHI boot loader and debugging, between USB (High) or COM1(Low). (refer to GHI boot loader section) This pin is high (select USB) if not connected.
71	VDD3			Connect to 3.3 volt source
72	VSS5			Connect to Ground
73	P2.2	IO1*	COM2	Serial port (UART) CTS signal for COM2
74	P2.1	IO3*	COM2	Serial port (UART) RXD receive signal (In) for COM2
75	P2.0	IO5*	COM2	Serial port (UART) TXD transmit signal (Out) for COM2
76	P0.9	IO6*	SPI2	SPI master bus interface MOSI signal (Master Out Slave In) for SPI2
77	P0.8	IO8*	SPI2	SPI master bus interface MISO signal (Master In Slave Out)for SPI2
78	P0.7	IO10*	SPI2	SPI master bus interface SCK signal (Clock)for SPI2
79	P0.6	IO12*		General purpose digital I/O
80	P0.5	IO14*	CAN2	TD CAN Channel 2 Data Transmit pin (Out)
81	P0.4	IO16*	CAN2	RD CAN Channel 2 Data Receive pin (In)
82	P4.28	IO13	COM4	Serial port (UART) TXD transmit signal (Out) for COM4
83	VSS7			Connect to Ground
84	VDD_3			Connect to 3.3 volt source
85	P4.29	IO17	COM4	Serial port (UART) RXD receive signal (In) for COM4
86	P1.17	IO60	N/A	General purpose digital I/O
87	P1.16	IO59	N/A	General purpose digital I/O
88	P1.15	IO58	N/A	General purpose digital I/O
89	P1.14	IO57	N/A	General purpose digital I/O

	Name			* Interrupt capable input.
No.	**LPC2387 H/W**	**USBizi IO**	**2nd Feature**	**USBizi Pin Description**
90	P1.10	IO56	N/A	General purpose digital I/O
91	P1.9	IO55	N/A	General purpose digital I/O
92	P1.8	IO54	N/A	General purpose digital I/O
93	P1.4	IO53	N/A	General purpose digital I/O
94	P1.1	IO52	N/A	General purpose digital I/O
95	P1.0	IO51	N/A	General purpose digital I/O
96	VDD4			Connect to 3.3 volt source
97	VSS6			Connect to Ground
98	P0.2	IO18*	COM1	Serial port (UART) TXD transmit signal (Out) for COM1
99	P0.3	IO20*	COM1	Serial port (UART) RXD receive signal (In) for COM1

Note: IO27,IO30, IO37, and IO70 are not supported in 100 package.

3. USBizi On Boot Up

USBizi includes two pieces of embedded software, GHI boot loader and USBizi firmware.

On system boot up, GHI boot loader initializes Flash and RAM memory then it looks for a valid USBizi firmware. USBizi firmware is the main software that runs .NET Micro Framework core and the user managed application.

During boot up a user can interrupt the sequence to remain in boot loader to update the USBizi firmware for instance.

3.1. GHI Boot Loader vs. USBizi Firmware

The following table lists the major properties of each software:

GHI boot loader	USBizi firmware
Used to update USBizi firmware or for low level USBizi flash maintenance.	Used to deploy, execute and debug the managed NETMF application code. In another words, it plays the role of a virtual machine.
Pre-burnt on the chipset's flash memory.	User download-able to USBizi chipsets, through GHI boot loader for instance.
Fixed and can not be updated.	Latest file is included with every GHI NETMF SDK.
Access interface can be USB or COM1 serial port on USBizi chipset.	Access Interface (debugging interface) can be USB or COM1 serial port.
User interface is a simple command line interface through any terminal service software such as TeraTerm or Hyper Terminal.	Users access it through Microsoft Visual C# to deploy, execute and debug the managed NETMF application through the debugging interface. Users can access it using Microsoft NETMF MFDeploy tool or APIs to maintain the firmware or application code region.
Very compact to accomplish the flash memory and firmware maintenance functions.	Highly sophisticated with .NET Micro Framework and requires HAL and PAL drivers to provide the various USBizi features.

Next sections provide more details.

3.2. Boot up Modes Settings and Access Interface Drivers

On system boot up USBizi checks BL and MODE pins states to choose between different modes as explained in the following table:

BL	MODE	Access Interface	Drivers and Interface Settings	User Interface Software	Mode
High or unconnected	High or unconnected	USB client	Driver: In GHI NETMF SDK, GHI NETMF Interface.	Microsoft MFDeploy tool and Visual C#.	USBizi firmware is executed if verified successfully then user application.
Low	High or unconnected	USB client *Virtual COM*	Driver: In GHI NETMF SDK, GHI Bootloader Interface.*(USB CDC device)*	Terminal Service software. Highly Recommended: TeraTerm available on www.ghielectronics.com	GHI Boot Loader.
High or unconnected	Low	Serial COM1	No driver Baud rate: 115200 Bus width: 8 bits Parity: none Stop bits: 1 bit Flow Control: none	Terminal Service software. Highly Recommended: TeraTerm available on www.ghielectronics.com	USBizi firmware is executed if verified successfully then user application.
Low	Low				GHI Boot Loader.

Thus, the default mode is USBizi firmware with USB debugging access interface.

Note: BL (IO0) and MODE (IO4) pins can still be used from managed code after USBizi firmware boots up.

4. GHI Boot Loader

USBizi Boot Loader is a software developed by GHI and is included on all USBizi chipsets. It is used to update the USBizi firmware or for low level USBizi flash maintenance.

GHI boot loader accepts simple commands in ASCII characters sent with help of a terminal service software (TeraTerm). Thus the user sends the desired command character and the boot loader performs an action. The results are returned in a human friendly format followed by a "BL" indicating that the boot loader is ready for the next command.

Boot up modes, settings and drivers section provides the required information on how to access GHI boot loader.

GHI boot loader is different than USBizi firmware, GHI boot loader vs. USBizi firmware section lists the features and properties of each piece of software.

4.1. GHI Boot Loader Commands

CMD	Description	Notes
V	Returns the GHI Loader version number.	Format *X.XX* *e.g. 1.06*
E	Erases the Flash memory (except the boot loader region).	Confirm erase by sending Y or any other character to abort. This command erases USBizi firmware and the user's application region.
X	Loads the new USBizi firmware.	USBizi firmware update section explains this command process in more details.
R	Runs firmware.	Exits GHI boot loader mode and forces running USBizi firmware.
L	Loads the managed application code.	Used to update the user's application from GHI boot loader. Usually used to load the application in production. Managed application in production stage section explains this command process in more details.
G	Reads the managed application code.	Saves the user's application flash memory region into a redistributable file. Managed application in production stage section explains this command process in more details.
D	Deletes the managed application code.	Confirm erase by sending Y or any other character to abort. This command only erases the user's application region. Rarely used to force deleting the application in some cases when MFDeploy or Visual C# tool can not access USBizi firmware interface due to some blocking caused by the user's application code.
P	Disables reading the managed application code.	This command blocks G command to protect the final product application from piracy.

4.2. USBizi Firmware Update

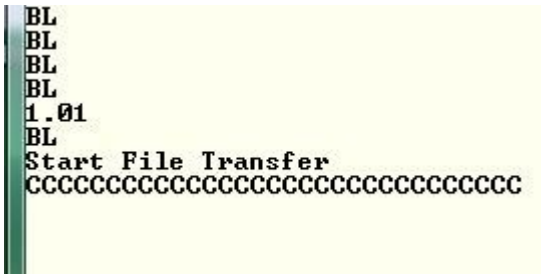
Always make sure the USBizi firmware is loaded on the device and the assemblies in your application in Visual C# have compatible version. Ideally, whenever you install a new SDK on your PC, you will also update the firmware on your device and update the assemblies you have added to your application.

First we want to make sure to start fresh before loading the new firmware:

1. Access the boot loader using TeraTerm as explained earlier.
2. Erase the flash memory using E command then press Y to confirm (this will take several seconds).
3. Loading new Firmware is simple but it requires a terminal that supports XMODEM file transfer. XMODEM has many versions, GHI boot loader requires 1K transfers with 16-bit CRC error checking. Keep on using TeraTerm software.

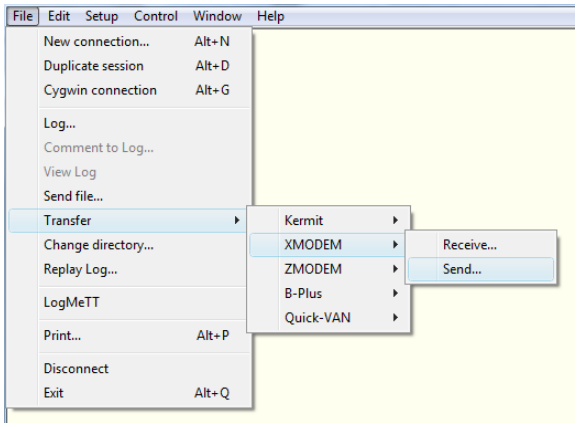
   Transfer is initiated using the X command. After the X command is entered, the GHI boot loader will start sending back the “C” character continuously. This “C” is an indicator that tells XMODEM a device is waiting for data. Once you see “C” character appearing on the terminal window, you can select XMODEM transfer and point the software to the firmware file from the SDK "USBizi_CLR.GHI".

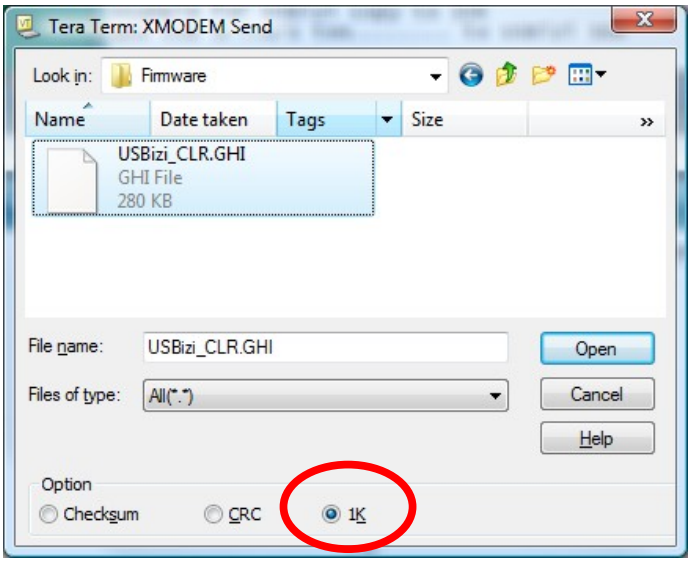
Entering X command:



In the menu, select **File > Transfer > XMODEM > Send...**

Next, select the firmware file from the SDK.





Updating the firmware takes a few seconds to load. Once loading has finished and the file is valid, the new firmware is executed automatically and you will not see “BL” again.

4.3. Managed (User) Application in Production Stage with Copy Protection

Updating your managed application from the GHI boot loader is available from boot loader version 1.06 and above. This is similar to updating USBizi firmware in the previous section, but instead it updates the managed application.

Also, the managed application can be protected from reading. This is useful if you need to protect your managed application against copying, tampering or disassembling.

Although extensive testing is done on this feature, GHI cannot guarantee or be held accountable for the possibility of hacking or bypassing protection.

Here are the expected steps in developing an application for USBizi:

1. Develop the application and deploy using Visual Studio.
2. When the application development is done, the user will read the application using the GHI boot loader G command. This is using XMODEM 1K Receive mode.
3. In production, the application is simply loaded using the L command and XMODEM 1K Send mode.
4. If read protection is needed, the user can use the P command.

5. USBizi Firmware

USBizi firmware is the main piece of embedded software in USBizi chipset which hosts .NET Micro Framework core with the required HAL drivers to provide the various USBizi features a user can control with C# developed managed application code. A user deploys and debugs the managed application code directly on USBizi chipsets from Microsoft's Visual Studio through USBizi debugging interface.

Boot up modes settings and drivers section provides the required information on how to access USBizi debugging interface.

USBizi firmware is different from GHI boot loader, GHI boot loader vs. USBizi firmware section lists the features and properties of each piece of software.

The end-user software interface that communicates with EMX firmware is MFDeploy, which comes with Microsoft .NET Micro Framework SDK and Microsoft Visual C# with installed .NET Micro Framework SDK.

5.1. Getting Started with USBizi

The objectives of this section is provide simple steps to setup your USBizi-based system on your PC, so you're ready to develop your application on Visual Studio C# with .NET Micro Framework.

The suggested access interface in these steps is USB (the default).

All you need to start up

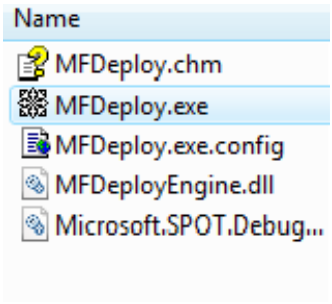
- USBizi-based system such as USBizi Development System, FEZ Domino, FEZ Mini or you custom hardware design.
- USB Cable.
- Microsoft Visual Studio 2010 or Microsoft Visual C# Express 2010 Edition (free download) installed with the latest updates.
- Microsoft .NET Micro Framework SDK Version 4.1.
- Latest GHI NETMF SDK, available on GHI Electronics website.

First Power-up

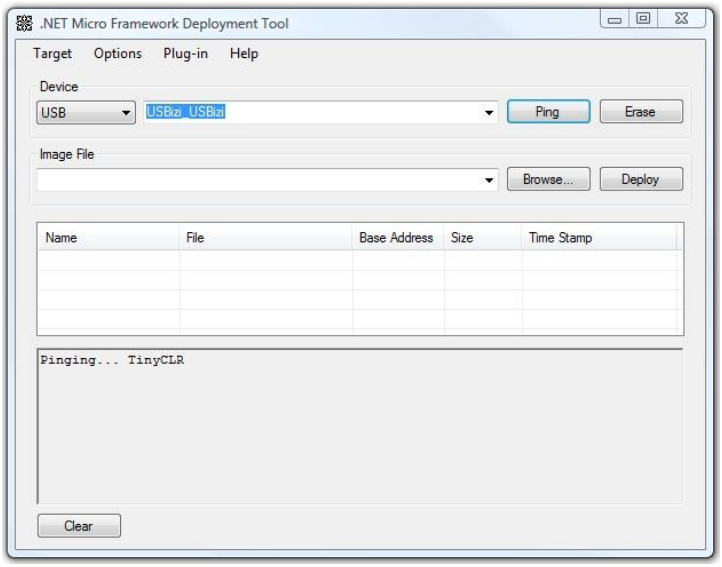
1. Install the latest Microsoft .NET Micro Framework SDK Version 4.1.
2. Install the latest GHI NETMF SDK.
3. Set BL=Low, MODE=High or keep it unconnected.

4. Power up the system (connect USB cable).
5. The system up will boot up in GHI boot loader mode with USB interface.
6. Update USBizi firmware.
7. Reset and boot up in USBizi firmware mode. (Leave BL and MODE pins High or floating on start-up). Lead Windows to the GHI NETMF Interface driver included with SDK.
8. Run the MFDeploy tool and choose USB from the device list then you'll see *USBizi_USBizi* in the list. *MFDeploy is a free tool with NETMF SDK that provides managed application code maintenance with USBizi.*

   %Microsoft .NET Micro Framework\v4.1 folder%\Tools\MFDeploy.exe



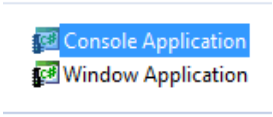
9. Press the “Ping” button on MFDeploy. It should return “TinyCLR”. This verifies that the board is responsive.



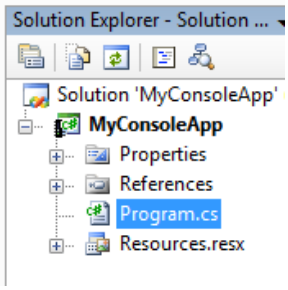
Note: If you did not see that string you may have a different default debugging interface (you might have installed the driver incorrectly, or the processor is

shutdown).

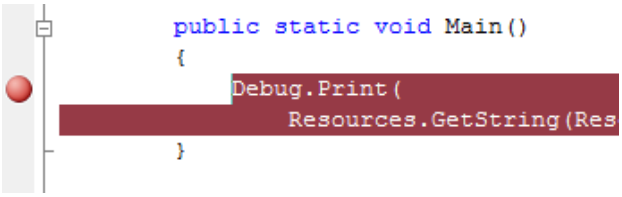
10. Open Visual Studio and start a new Micro Framework project with the “Console Application” template. This is the simplest application that can be loaded. All it does is print a string to the debug output. Name your project “MyConsoleApp.”



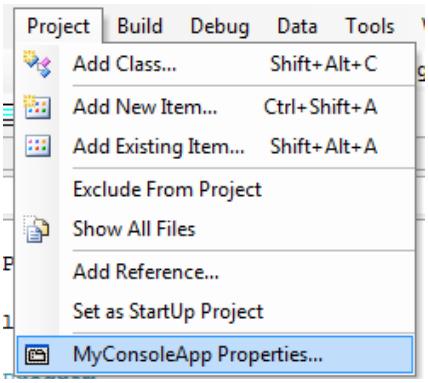
11. Visual Studio will now generate all the needed project files. One of the files is called *Program.cs*, open it...

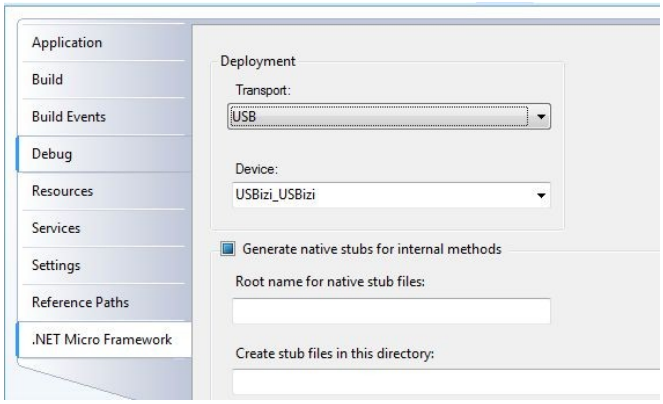


12. Place a breakpoint on the Debug.Print line. You can do this by clicking on the line and then pressing F9.

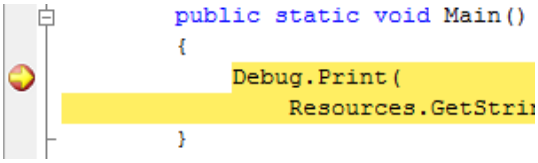


13. Compile the application. There should be no errors.
14. Go to the menu and select **Project > MyConsoleApp Properties...** and in the new window select the “.NET Micro Framework” tab. In the tab, there are options for deployment. Select *USB* from the Transport drop-down and select *USBizi_USBizi* from the Device drop-down.

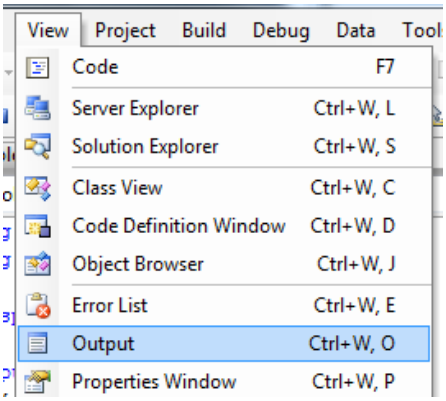




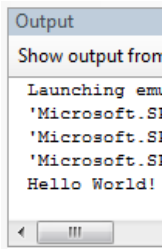
15. Press F5 (Debug). You will see how Visual Studio loads the application and runs it. Visual Studio should pause at the breakpoint we placed in step 4.



16. Make sure you have the Output window open. If not, you can open the Output window from **View > Output**.



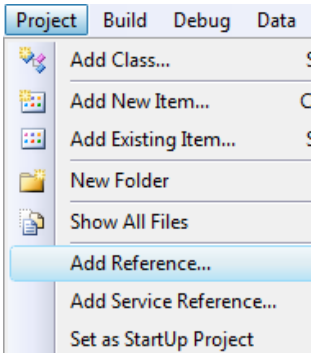
17. Press F10 to step over Debug.Print and watch the Output window. The Output window should display “Hello World!”



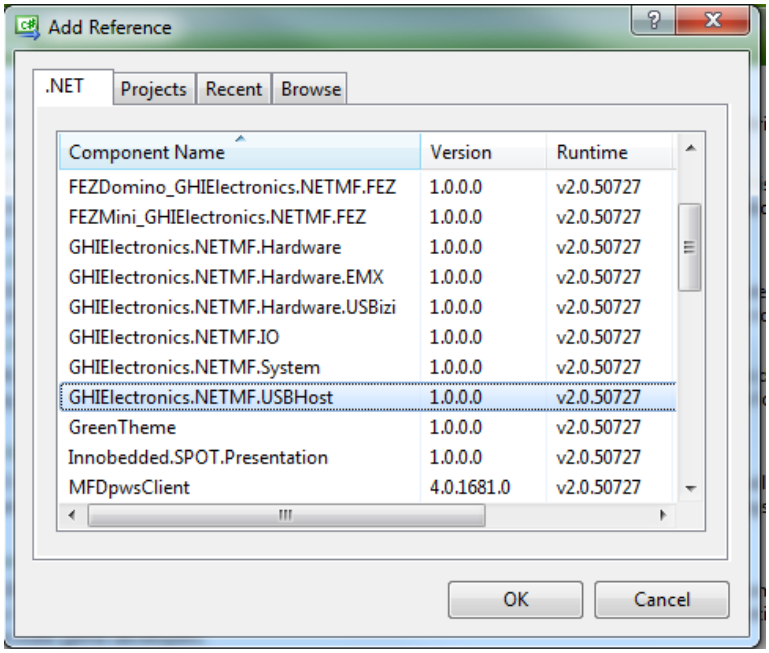
18. Press F5 and the code will continue executing until it reaches the end of the program.

Adding GHI NETMF Library

1. Go to the **Project** tab and click **Add Reference**.



2. Let's add USB Host library. Select it and click OK.



3. Add a using for the name space at the beginning of the file:

   ```
   using GHIElectronics.NETMF.USBHost;
   ```

4. As an example, we will get a list of currently connected devices. Add this in Main() method:

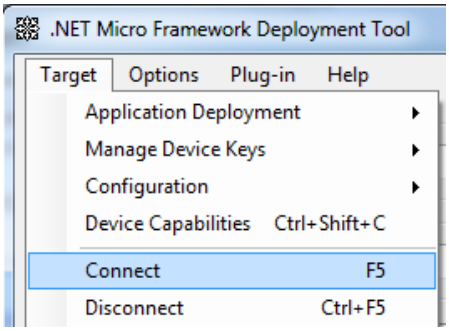
   ```
   USBH_Device[] devices = USBHostController.GetDevices();
   ```

5. Similarly, you can use any other functionality provided by GHI library. Press F5 in visual studio and the program will run.

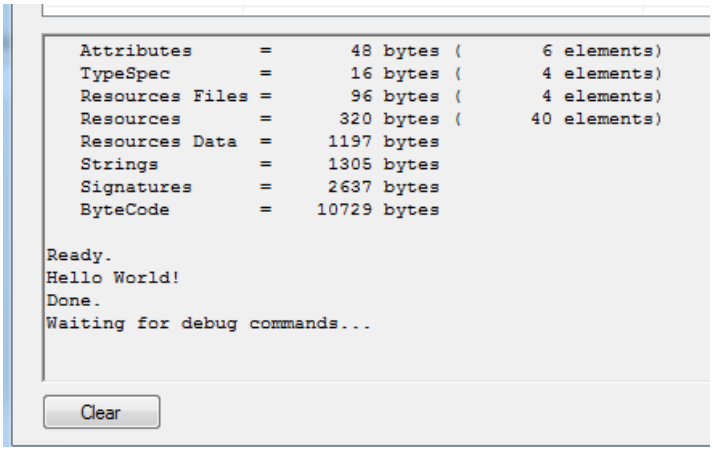
   If the program does not run, then there is something incompatible on your system. For example, your are using a newer or incorrect version of the GHI library and older or incorrect version of the firmware is running on your hardware. This is simply resolved by upgrading the firmware to the one included in your SDK and making sure the Added Reference is from the SDK as well.

   MFDeploy is helpful to investigate these errors as explained next.

Using MFDeloy, you can see any debug messages, exceptions or errors from your device. Make sure Visual Studio is not in debug mode or close it. Open MFDeploy and make sure you can ping as explained in previous steps. Now, Click on **Target->Connect**.



Now, reset your hardware and click ping. You will see debug output of what the device is doing, for example loading assemblies and any debug messages printed by your application.


Attributes = 48 bytes (6 elements)
TypeSpec = 16 bytes (4 elements)
Resources Files = 96 bytes (4 elements)
Resources = 320 bytes (40 elements)
Resources Data = 1197 bytes
Strings = 1305 bytes
Signatures = 2637 bytes
ByteCode = 10729 bytes
Ready.
Hello World!
Done.
Waiting for debug commands...
Clear

In case the program did not run because of incompatibility, the debug output will show these errors. This is useful for debugging certain applications.

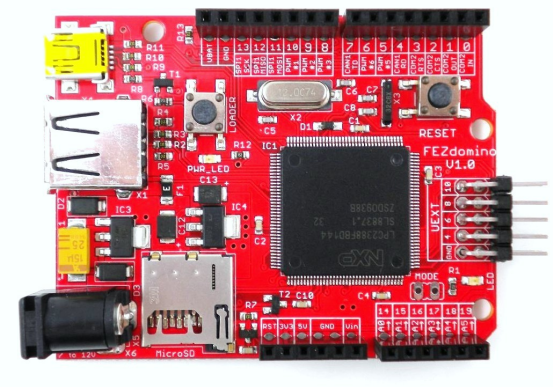
Note: If you Connect through MFDeploy, you cannot deploy using Visual Studio anymore. MFDeploy must be disconnected or closed first and then you can go back to Visual Studio. Only one of these two programs can be connected to your platform at one time.

5.2. Recommended Hardware and Tutorials for USBizi

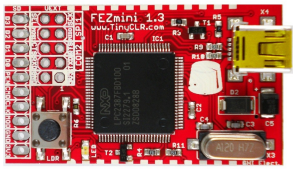
GHI Electronics offers an open source hardware development platform for beginners powered by USBizi.

From the ground up, FEZ Mini and FEZ Domino boards are designed with help of USBizi chipsets to be extremely easy to learn. For more details about these products visit www.tinyclr.com

This website also offers a free eBook for .NET Micro Framework beginners in addition to the FEZ (USBizi) tutorials.



FEZ Domino Based on USBizi LQFP144



FEZ Domino Based on USBizi LQFP100

6. USBizi Features

USBizi firmware supports all the necessary features of .NET Micro Framework version 4.1 with all the required HAL and PAL drivers such as FAT File System. .NET Micro Framework SDK includes full documentation and examples about the usage of these features with the related libraries.

Furthermore, USBizi supports other exclusive GHI hardware and software features such as USB host, PWM, ADC and DAC. The SDK includes the required library files with full documentation and examples about the usage of these features with the related libraries.

The following sections clarify necessary guidelines about USBizi features.

6.1. Application Flash/RAM

Since USBizi firmware is implemented on a single-chip, it depends on the available memory resources embedded in the chip, which is enough for wide range of tiny embedded systems.

The available memory should be used wisely. For example, all unused objects in the code should be disposed.

RAM	96 KB
Flash	512 KB
User available RAM	About 60 KB
User available Flash	About 150 KB

6.2. Debugging Interface (Access Interface)

This is the communication interface between USBizi firmware and the application code terminal (Visual C# debugger). It can be configured as USB or serial port (COM1 on USBizi hardware).

Boot up modes settings and access interface drivers section provides the required information on how to access USBizi debugging interface.

6.3. Digital Inputs/Outputs

All Digital IO pins are 3.3V and 5V tolerant. This means that signals coming from another circuit can be up to 5V (e.g. connecting USBizi to a 5V microcontroller).

All pins support input and output with pull up and pull down.

Refer to Pin-Out Description section for more information about Digital IOs assignment to USBizi hardware pins.

Most digital I/O pins are interrupt capable. Interrupt pins asynchronously call functions in managed applications. Interrupts can be activated on rising or falling edges with an optional glitch filter. Enabling interrupts for both rising and falling edges is supported but in this case the glitch filter is disabled. Interrupt capable pins are marked in the pin-out table.

Important Note: Inputs are 5V tolerant but USBizi can not be powered by 5 volts.

6.4. Serial Peripherals

Serial Port (UART)

One of the oldest and most common protocols is UART (or USART). USBizi hardware exposes 4 UART ports

Serial Port	LPC2388 or LPC2387 UART	Hardware Handshaking
COM1	UART0	Not Supported
COM2	UART1	Supported
COM3	UART2	Not Supported
COM4	UART3	Not Supported

Important Note: Serial Port pins have 3.3V TTL levels where the PC uses RS232 levels. For proper communication with RS232 serial ports (PC serial port), an RS232 level converter is required. One common converter is MAX232. If the serial port is connected between two TTL circuits, no level converter is needed but they should be connected as a null modem. Null modem means RX on one circuit is connected to TX on the other circuit, and vice versa.

Refer to Pin-Out Description section for more information about UART signals assignment to USBizi hardware pins.

SPI

USBizi supports two SPI interfaces, SPI1 and SPI2. SPI Bus is designed to interface multiple SPI slave devices, the active slave is selected by asserting Chip Select line on

the relative slave device.

Refer to Pin-Out Description section for more information about SPI signals assignments to USBizi hardware pins.

I2C

I2C is a two-wire addressable serial interface. USBizi supports one master I2C port.

Refer to Pin-Out Description section for more information about I2C signals assignments to USBizi hardware pins.

CAN

Controller Area Network is a common interface in industrial control and automotive. CAN is remarkably robust and works well in noisy environments. All error checking and recovery methods are done automatically on the hardware. TD (Transmit Data) and RD (Receive Data) are the only pins needed. These pins carry out the digital signals that need to be converted to analog before it can be used. There are different CAN transceivers. The most common one is dual-wire high speed transceivers, capable of transferring data up to 1MBit/second.

Refer to Pin-Out Description section for more information about SPI signals assignments to USBizi hardware pins.

This is available through GHI NETMF library.

One-wire Interface

Through one-wire a master can communicate with multiple slaves using a single digital pin. One-wire can be activated on any Digital I/O on EMX.

This is available through GHI NETMF library.

6.5. Networking

Ethernet

USBizi supports Ethernet Networking through Wiznet. Still under development, more details are coming soon.

6.6. Graphics

USBizi does not have native graphics support. However, using an LCD with an SPI based graphics accelerator, such as SSD1339 LCD controller, you could easily add this feature to USBizi.

6.7. USB Client (Device)

USB Client (device) *and* ***USB Host*** *are completely different. Many designers confuse USB when it comes to host and device. USB Host is the master of the bus where all the work is done. USB devices are simple compared to host and they can only connect/communicate with a host and not other devices. USB host and device on EMX are two separate peripherals, so there would be no conflict when using them both simultaneously.*

The USB client interface is usually used as an USBizi access interface for debugging and application deployment through Microsoft Visual Studio. However, developers have full control over the USB client interface. For example, the USB client can be made to simulate a USB keyboard or USB mass storage.

Controlling a USBizi USB client requires intricate knowledge of how USB works. The user should refer to .NET Micro Framework documentation for complete details on how to use this feature.

Fortunately, GHI Electronics offers a USB Client library (available in the SDK) to ease development and provide direct support for some USB devices, such as, Mass Storage (Virtual Disk) and CDC (Virtual COM Port). The library is capable of creating a USB client that's composed of multiple USB interfaces. Please refer to GHI NETMF Library for more information.

Usually, devices support USB host or device. USBizi LQFP144 package contains both of them and it is up to the final application to have both or one of them. USBizi LQFP100 package supports USB device only.

Refer to Pin-Out Description section for more information about USB device signals assignment to USBizi hardware pins.

Important Notes:

- **Be CAREFUL when changing the USB configuration and settings**, as you go on with development and creating your USB device and connecting it to the PC, Windows might save the device information in its registry. Therefore, if you change the USB device settings/interfaces and connect it again, it might not work correctly. Make sure to be careful with changing your USB device settings. You may also need to delete all the settings from Windows registry manually.
- By default, Micro Framework debug interface is USB. If you need to use the USB Client feature to build a USB device, you should select a different debug interface first (COM1).
- Make sure to select 64 bytes as the **bMaxPacketSize0** in the Device Descriptor.
- USBizi uses LPC2388/LPC2387 as the core processor which has a fixed endpoint configuration and the user must comply with these restrictions, otherwise the USB device configuration will be refused by USBizi. Here's a table of how the endpoints are assigned: (LPC23xx data sheet has complete reference).

Endpoint Number	Endpoint Type	Direction	Double Buffer
0	Control	In/Out	No
1	Interrupt	In/Out	No
2	Bulk	In/Out	Yes
3	Isochronous	In/Out	Yes
4	Interrupt	In/Out	No
5	Bulk	In/Out	Yes
6	Isochronous	In/Out	Yes
7	Interrupt	In/Out	No
8	Bulk	In/Out	Yes
9	Isochronous	In/Out	Yes
10	Interrupt	In/Out	No
11	Bulk	In/Out	Yes
12	Isochronous	In/Out	Yes
13	Interrupt	In/Out	No
14	Bulk	In/Out	Yes
15	Bulk	In/Out	Yes

USB Cable Connection Detection

USB VBUS can be connected, through a protection resistor, to any digital IO to detect the presence of a USB cable.

6.8. USB Host and Supported Class Drivers

USB Client (device) *and* ***USB Host*** *are completely different. Many designers confuse USB when it comes to host and device. USB Host is the master of the bus where all the work is done. USB devices are simple compared to host and they can only connect/communicate with a host and not other devices. USB host and device on EMX are two separate peripherals, so there would be no conflict when using them both simultaneously.*

USB Host allows the use of USB Hubs, USB storage devices, joysticks, keyboards, mice, printers and more. With EMX supported class drivers, you don't have to worry about the inner workings. For USB devices that do not have a standard class, low level USB access is supported.

Usually, devices support USB host or device. USBizi LQFP144 package contains both of them and it is up to the final application to have both or one of them. USBizi LQFP100 package supports USB device only.

Refer to Pin-Out Description section for more information about USB Host signals assignment to USBizi hardware pins.

This is available through GHI NETMF library.

6.9. Storage Devices (SD, USB MS) / File System

File System lets you create and manipulate files and folders on the connected SD and USB storage devices.

With Micro Framework V4.1, FAT32 and FAT16 are supported by NETMF. The user should refer to .NET Micro Framework documentation for details on handling files and folders.

Note: FAT32 and FAT16 formats are supported, but FAT12 is not. You can format your storage device on a PC with a FAT32 or FAT16 option before using on USBizi.

Before using the storage devices and accessing them with NETMF, the user must mount the file system first. This is done using the USBizi library provided with the SDK. SD cards and USB storage devices are **NOT** mounted automatically.

Please refer to library documentation: GHIElectronics.NETMF.IO.PersistentStorage

SD/MMC Memory

SD and MMC memory cards have similar interfaces. USBizi supports both cards and also supports SDHC (over 2GB) cards. The interface runs at 4Bits when using SD cards and 1Bit when using MMC cards.

There are two smaller versions of SD cards, mini SD and micro SD. All three card sizes are identical as far as the interface. All card sizes work with USBizi.

Refer to Pin-Out Description section for more information about SD signals assignment to USBizi hardware pins.

A user might be interested in mounting or unmounting the file system on the SD card automatically when a SD card is inserted or ejected. To do this, there is a pin on the SD card connector called Card Detect which works like a switch. Connect this to a digital I/O InterruptPort on USBizi and call mount or unmount appropriately.

USB Mass Storage

USB mass storage devices such as USB hard drives or memory sticks are directly supported on USBizi.

Please refer to library documentation: GHI Electronics.NETMF.IO.PersistentStorage

6.10. Analog Inputs/Outputs

Analog inputs can read voltages from 0V to 3.3V with 10Bit resolution. Similarly, The analog output can set the pin voltage from 0V to 3.3V (VCC to be exact) with 10Bit resolution.

Although the pins are 5V tolerant, the ADC multiplexing is not and this can cause wrong readings on the affected pin or other analog pins. Please consult LPC23xx user manual for more details.

Refer to Pinout Description section for more information about Analog input/output assignments to USBizi hardware pins.

This is available through GHI NETMF library.

6.11. PWM

The available PWM pins have built-in hardware to generate the signals. No resources are needed to generate PWM.

Note that all PWM pins share the same timer. Changing one PWM frequency will affect the others.

This is available through GHI NETMF library.

6.12. Output Compare

Using output compare developers can generate different waveforms. This is available on any digital output pin.

This is available through GHI NETMF library.

6.13. Battery RAM

EMX has 2KB of RAM that is backed-up by battery. Data is retained on power loss. The developer needs to wire a 3V battery or a super capacitor to the VBAT pin.

This is available through GHI NETMF library.

6.14. Power Control / Hibernate

Power Control

USBizi is running at 72MHz. Different low power modes are possible.

This feature is still under development.

Hibernate

Hibernate is supported to save power. The processor will go to sleep and wakeup on specific events.

This is available through GHI NETMF library.

6.15. Real Time Clock

LPC2388 or LPC2387 includes a real time clock that can operate while the processor is off. The developer needs to wire a battery or a super capacitor to VBAT pin and 32.768 KHz crystal to RTCX1 and RTCX2 pins. RTC also provides alarm functionality.

This is available through GHI NETMF library.

6.16. Processor Register Access

USBizi Chipset allows direct access to the LPC2388 or LPC2387 registers. The user can write, read or manipulate the bits as needed. This can be useful, enabling some features that may not be already exposed.

This is available through GHI NETMF library.

6.17. Managed Application Protection

With USBizi you can disable reading the deployed application memory region. This is useful if you need to protect your managed application against copying, tampering or disassembling. This functinality was added to GHI boot loader.

6.18. Watchdog

Watchdog is used to reset the system if it enters an erroneous state. The Watchdog is enabled with a specified timeout. The user must keep resetting the Watchdog time counter within this timeout interval or otherwise the system will reset.

This is available through GHI NETMF library.

7. Advanced Users

USBizi Chipsets are based on NXP LPC2388 and LPC2387 microcontrollers. With USBizi firmware's register access feature, advanced users, familiar with NXP microcontrollers, can manipulate the internal chipset's register which provided the ultimate flexibility.

For example, a special design requires remapping COM4 (UART3) RX and TX pins from P4.29 from P2.29 to P0.26 and P0.25:

```
// add this on top of your code
using GHIElectronics.NETMF.Hardware.LowLevel;
using System.IO.Ports;
//...
//...
// add this function anywhere
static public void RemapCOM4(SerialPort ser)
 {
    // call this function **after** you open COM4 port
    if (ser.PortName != "COM4" || ser.IsOpen == false)
        throw new Exception("Only use COM4 and make sure it is open");
    // remap COM4 RX (in)  pin from P4.29 to P0.26
    // remap COM4 TX (out) pin from P4.28 to P0.25
    Register PINSEL9 = new Register(0xE002C024);
    PINSEL9.Write(0);// COM4 is now disconnected from P4.28 and P4.29
    Register PINSEL1 = new Register(0xE002C004);
    PINSEL1.SetBits(0xf << 18);// COM4 is now connected to An3 and An4
 }
```

8. USBizi Design Consideration

For final designs, the user should make sure to expose needed pins for USBizi functionality. MODE and BL pins are used to access the boot loader and change the debug interface. Also, the used debug interface (USB by default or serial) should be available. This is necessary to to load and update the managed application and firmware.

USBizi chipset needs 12MHz crystal for operation. Use GHI development boards schematics as a reference. For other details, please refer to LPC23xx user manual.

Legal Notice

Licensing

EMX Module is fully licensed for commercial use. The Module price covers the commercial use of EMX Module with .NET Micro Framework.

Disclaimer

IN NO EVENT SHALL GHI ELECTRONICS, LLC. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS PRODUCT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT ANY NOTICE. GHI ELECTRONICS, LLC LINE OF PRODUCTS ARE NOT DESIGNED FOR LIFE SUPPORT APPLICATIONS.

EMX is a Trademark of GHI Electronics, LLC

.NET Micro Framework, Visual Studio, MFDeploy, Windows Vista, Windows SideShow are registered or unregistered trademarks of Microsoft Corporation.

Other Trademarks and Registered Trademarks are Owned by their Respective Companies.